DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Subject Name: **ARTIFICIAL INTELLIGENCE** Subject Code: **CS T71**

UNIT II

Knowledge Representation: Approaches and issues in knowledge representation-Propositional Logic –Predicate logic-Forward and backward reasoning - Unification-Resolution- Weak slot-filler structure – Strong slot-filler structure- Knowledge- Based Agent

11 Marks

KNOWLEDGE REPRESENTATION IN AI (Nov'13)(Apr-May'14)

Knowledge:

- Is a fact or more than that?
- May be declarative or procedural.
- Procedural knowledge is compiled knowledge related to the performance of some task. E.g. Steps used to solve an algebraic equation
- Declarative knowledge is passive knowledge expressed as statements of facts about the world. E.g. Personnel data in a database

Knowledge Representation and Reasoning

Intelligent agents should have capacity for:

- **Perceiving**, that is, acquiring information from environment,
- **Knowledge Representation**, that is, representing its understanding of the world,
- **Reasoning**, that is, inferring the implications of what it knows and of the choices it has, and
- **Acting**, that is, choosing what it want to do and carry it out.

Representation of knowledge and the reasoning process are central to the entire field of artificial intelligence. The primary component of a knowledge-based agent is its knowledge-base. A knowledge-base is a set of sentences.

Each sentence is expressed in a language called the knowledge representation language. Sentences represent some assertions about the world. There must mechanisms to derive new sentences from old ones. This process is known as inferencing or reasoning.

Inference must obey the primary requirement that the new sentences should follow logically from the previous ones. *Logic* is the primary vehicle for representing and reasoning about knowledge. Specifically, we will be dealing with formal logic.

The advantage of using formal logic as a language of AI is that it is precise and definite. This allows programs to be written which are declarative - they describe what is true and not how to solve problems. This also allows for automated reasoning techniques for general purpose inferencing. This, however, leads to some severe limitations. Clearly, a large portion of the reasoning carried out by humans depends on handling knowledge that is uncertain. Logic cannot represent this uncertainty well.

Similarly, natural language reasoning requires inferring hidden state, namely, the intention of the speaker. When we say, "One of the wheel of the car is flat." we know that it has three wheels left. Humans can cope with virtually infinite variety of utterances using a finite store of commonsense knowledge.

Formal logic has difficulty with this kind of ambiguity.

– A logic consists of two parts, a language and a method of reasoning. The logical language, in turn, has two aspects, syntax and semantics. Thus, to specify or define a particular logic, one needs to specify three things:

– **Syntax:** The atomic symbols of the logical language, and the rules for constructing well-formed, non-atomic expressions (symbol structures) of the logic. Syntax specifies the symbols in the language and how they can be combined to form sentences. Hence facts about the world are represented as sentences in logic.

- Semantics: The meanings of the atomic symbols of the logic, and the rules for determining the meanings of non-atomic expressions of the logic. It specifies what facts in the world a sentence refers to. Hence, also specifies how you assign a truth value to a sentence based on its meaning in the world. A fact is a claim about the world, and may be true or false.

- Syntactic Interference Method: the rules for determining a subset of logical expressions, called theorems of the logic. It refers to mechanical method for computing (deriving) new (true) sentences from existing sentences.

- **Facts** are claims about the world that are True or False, whereas a **representation** is an expression (sentence) in some language that can be encoded in a computer program and stands for the objects and relations in the world. We need to ensure that the representation is consistent with reality, so that the following figure holds:

- entails Representation: Sentences -------------> Sentences | | | | | Semantics | Semantics | refer to | refer to | | \/ follows \/ World: Facts ----------------- > Facts

There are a number of logical systems with different syntax and semantics. We list below a few of them.

- Propositional logic

- All objects described are fixed or unique

"John is a student" student (john) Here John refers to one unique person.

- First order predicate logic

- Objects described can be unique or variables to stand for a unique object "All students are poor". For All(S) [student(S) -> poor(S)]

  Here S can be replaced by many different unique students. This makes programs much more compact:

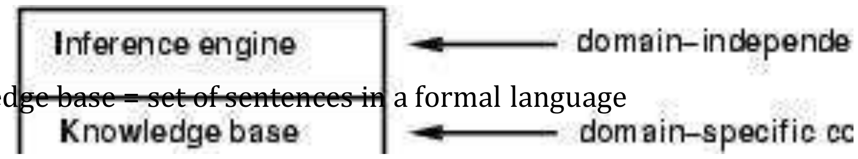
  E.g. for All (A, B) [brother (A, B) -> brother (B, A)]

- replaces half the possible statements about brothers
- Temporal
- Represents truth over time.
- Modal
- Represents doubt
- Higher order logics
- Allows variable to represent many relations between objects
- Non-monotonic
- Represents defaults
- Propositional is one of the simplest systems of logic.

2. What is Knowledge based agents. Explain

K<u>nowledge based systems</u>: systems that depend on rich base of knowledge to perform difficult tasks.

A knowledge based agent can combine general knowledge with current percepts to infer hidden aspects of the current state prior to selecting actions. Eg. Physician diagnoses a patient. The central component of a knowledge-based agent is its knowledge base. There must be a way to add new sentences to the knowledge base and a way to query what is known. Inference is deriving new sentences from old.



- Knowledge base = set of sentences in a formal language

- Declarative approach to building an agent (or other system):

- Tell it what it needs to know

- Then it can ask itself what to do - answers should follow from the KB

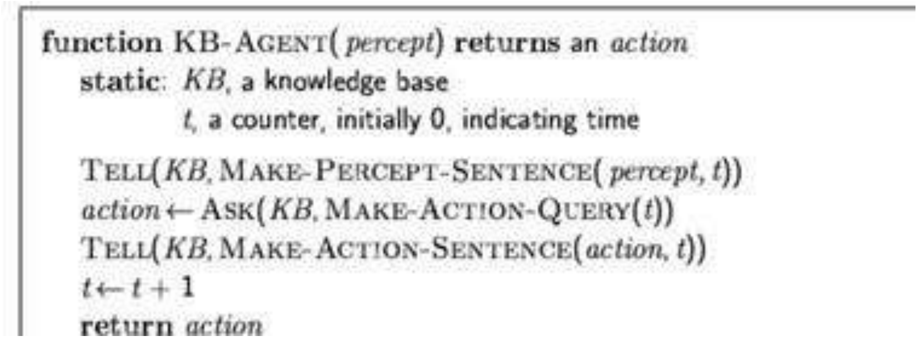
- Agents can be viewed at the knowledge level

  i.e., what they know, regardless of how implemented

- Or at the implementation level

– i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent



```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

Entailment

- Entailment means that one thing follows from another: KB $\models$ α
- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true
- E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"
- E.g., x+y = 4 entails 4 = x+y
- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

Inference

- *KB* $\vdash_i$ α = sentence α can be derived from *KB* by procedure *i*
- Soundness: *i* is sound if whenever *KB* $\vdash_i$ α, it is also true that *KB* $\models$ α

– Completeness: *i* is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

– Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

– That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

3. DISCUSS PROPOSITIONAL LOGIC? (APR-MAY'15)

Propositional logic is the simplest logic – illustrates basic ideas

Symbols:

The symbols of propositional logic are the logical contacts, true and false.

Eg: P, Q

Connectivities:

1. ∧ (and) : Conjunction

   Eg: P^Q

2. ∨(or) : Disjunction Eg: P∨Q

3 => (or) → (Implies): Implication or Conditional

Eg: $(P \wedge Q) \Rightarrow R$

$(P \wedge Q)$ – Premise or

antecedent R –

Conclusion or

consequent

4 $\Leftrightarrow$ (equivalent) – Equivalence or Biconditional

Eg: $(P \wedge Q) \Leftrightarrow (Q \wedge P)$

5. 7 (not) – Negotion is the only connective that operates on a single sentence.

Eg: 7P

A BNF grammar of sentences in propositional logic

Sentence $\rightarrow$ Atomic Sentence and complex sentence

Atomic sentence $\rightarrow$ True/False/P/Q/R/

Complex Sentence $\rightarrow$ (Sentence)

/sentence connective sentence

/ 7 sentence

Connective $\rightarrow$ ∧/∨/$\Leftrightarrow$ /=>

Order of Procedure: (from highest to lowest)

7, ∧, ∨, => and $\Leftrightarrow$

Eg: 7P∨Q∧R => S is equivalent to ((7P) ∨) Q∧R)) => S

TRUTH TABLE:

P	Q	7P	P∧Q	P∨Q	P=>Q	P $\Leftrightarrow$ Q
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	T	T	T	T

Rules of inference for Propositional Logic:

α $\vdash$ β (or) α/β – This notation represents that β can be derived from α by inference. The propositional logic has seven inference rules.

They are:

i). Moclus Ponens (or) Implication – Elimination:

$$\frac{\alpha => \beta, \alpha}{\beta}$$

ii). And – Elimination:

$$\frac{\alpha1\wedge\alpha2\wedge\ldots\ldots\wedge\alpha n}{\alpha1}$$

iii). And – Introduction:

$$\frac{\alpha1,\alpha2\ldots\ldots\ldots\alpha n}{\alpha1\wedge\alpha2\wedge\ldots\ldots\wedge\alpha n}$$

iv). or – Introduction:

$$\frac{\alpha1}{\alpha1,\vee\alpha2\vee\ldots\vee\alpha n}$$

v). Double – Negation Elimination:

From a double negation positive sentence is inferred $\frac{\prod\alpha}{\alpha}$

vi). **Unit Resolution:**

From a disjunction, if one of the disjoints is false, then the true one is inferred.

$$\frac{\alpha\vee\beta,7\beta}{\alpha}$$

vii). **Resolution:**

Implication is transitive

$$\frac{\alpha\vee\beta,7\beta\vee\gamma}{\alpha\vee\gamma}\qquad(or)\qquad\frac{7\alpha =>}{7\alpha}$$

4. What is Predicate logic OR FIRST ORDER LOGIC – FOL. Explain Forward and backward (Nov-Dec'14)(No'15)(Apr-May'17) (APR-MAY'16)

The proposition logic has a very limited ontology that is fact in the world and simple sentences are represented.

FOL makes a stronger set of ontological and functions of the world belongings to be represented.

1. Objects: Things with individual identities
2. Properties: Used to distinguish one object from other
3. Relations: Relation exist between objects
4. Functions: One kind of relation which has only one value for a given input.

<u>SYNTAX AND SEMANTICS of FOL:</u>

Terms: First order logic has sentence, but it also has terms which represents objects. Constant symbols, variables and function symbols are used to build terms and quantifiers and predicate symbols are used to build sentences.

SYNTAX of FOL in BNF (Backus – Naus Form):

Sentence $\rightarrow$ Atomic sentence

|Sentence connective sentence

|Quantifier variable,….sentence

7| sentence

|(sentence).

Atomic sentence → Predicate (Term,)

Term → Term

Term → Function (Term…)

|constant

|variable

Connective → =>|V|Λ| ⇔

Quantifier → ∀ ∃ |

Constant → A | λcon| john|

Variable → a| x |s |…..

Predicate → Before| Hascolour |Raining

Function → Mother| Left-hand of|……..

1. **Constants:** Each constant symbol denotes exactly one object not necessarily one name, might be represented by several names.

2. **Variables:** Assumes different valves in the domain during the inference technique.
3. **Predicate symbols:** It refers to a particular relation in the model.
4. **Tuple:** In a model the relation is defined by the set of tuple of objects. Tuple is a collection of objects arranged in a fixed order.
5. **Function symbol:** A type of relation that is exactly related to one of the other objects by the relation.
6. **Atomic Sentences**: An atomic sentence is formed from a predicate symbol.
7. **Complex sentence:** Logical connectives are used to construct move complex sentence in which the semantic of given sentence has to be satisfied.

QUANTIFIERS:

Quantifiers are used to express properties of entire collection of objects, rather than represent the object by names. FOL contains two standard quantifiers,

# Universal quantifier ($\forall$)

# Existential quantifier ($\exists$)

U NIVERSAL QUANTIFIERS:

General Notation: “$\forall X$ P” where,

P – Logical expression, X – Variable, $\forall$ - For all

That is, P is true for all objects X in the Universe.

Examples:

All cats are mammals => $\forall$ X cat (X) $\rightarrow$ mammals(X)

That is, all the cats in the universe belongs to the type of mammals and hence the variable X may be replaced by any of the cat name (object, Name)

Examples:

Spot is a cat

Spot is a mammal

Cat (spot)

Mammal (spot)

Cat (spot) $\rightarrow$ mammal (spot)

Spot – Name of the cat.

E<u>xistential Quantifiers:</u>

General Notification: $\exists$ X P, where

P – Logical Expression, X – Variable , $\exists$ - There exist

That is P is true for some object X in the universe.

Example:

Spot has a sister who is a

cat. $\exists$ X sister

($X, spot$) $\rightarrow$ cat(X

)

That is, the spot's sister is a cat, implies spot is also a cat and hence X may be replaced by, sister of spot, if it exists.

Example:

Felix is a cat.

Felix is a sister of spot

Cat (Felix)

Sister (Felix, spot)

Sister (Felix, spot) $\rightarrow$ cat (Felix).

<u>NESTED QUANTIFIERS:</u>

The sentences are represented using multiple quantifiers.

Example:

# For all X and all Y, if x is the parent of Y then Y is the child of X. $\forall$ X, Y parent (X, Y) $\rightarrow$ child (Y, X).

# Everybody loves somebody $\forall X \exists$ Y loves (X, Y)

# There is someone who is loved by everyone Y $\exists$ X $\forall$ loves (X, Y)

Connection between $\forall$ and $\exists$?

The two Quantifiers ($\forall$ and) $\exists$ are actually connected with each other through negation.

Example:

Everyone likes ice cream ∀X likes (X, ice cream) is equivalent to 7 ∃ X 7 likes (X, ice cream)
That is there is no one who does not like ice cream.

Ground Term or Clause: A term with no variable is called ground term. Eg: cat (spot)

The De Morgan rules for quantified and unquantified sentences are as follows,

# **Quantified sentences:**

∀X7P ≡ 7∃XP 7∀XP ≡ ∃X7P

∀XP ≡ 7∃ X7P

∃XP ≡ 7∀ X7P

#Unquantified sentence:

7(P∧Q) ≡ 7P ∨ 7Q

7P ∧ 7Q ≡ 7(P∨Q)

P ∧ Q ≡ 7(7P ∨ 7Q)

P∨ Q ≡ 7(7P ∧ 7Q)

5. What is UNIFICATION? Explain (Apr-May'15)

The require findings substitutions that make different logical expression. This proves is called unification and is a key component of all first order inference algorithms. The UNIFY algorithm takes two sentences and returns a unifier for them if one exists:

Unify (P,Q) = θ where SUBSET (θ,P)= SUBSET (θ, q)

Here are the results of unification with four different sentences that might be in knowledge base.

UNIFY (knows (john,x), knows (john,jane)) = {x|jane}

UNIFY (knows (john,x), knows (y,bill)) = {x|bil,y|john}

UNIFY (knows (john,x), knows (y,mother(y))) = {y|john,x|mother(john)}

UNIFY (knows (john,x), knows (x,Elizabeth)) = fail.

The last unification fails because x cannot take on the values john and Elizabeth at the sametime. Now remember that knows (x,Elizabeth) means "everyone knows Elizabeth", so we should be able to infer that john knows Elizabeth. The problem arises only because the two sentences happen to use the same variable name, x. the problem can be avoided by standardizing apart one of the two sentences being unified, which means remaining its variable to avoid name clashes

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

  θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p q θ Knows(John,x)

Knows(John,Jane) Knows(John,x)

Knows(y,OJ) Knows(John,x)

Knows(y,Mother(y))

Knows(John,x)   Knows(x,OJ)

– Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

– We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

– Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

– Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

– We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

– Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

– Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

– We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

– Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	

– Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

– We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

– Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

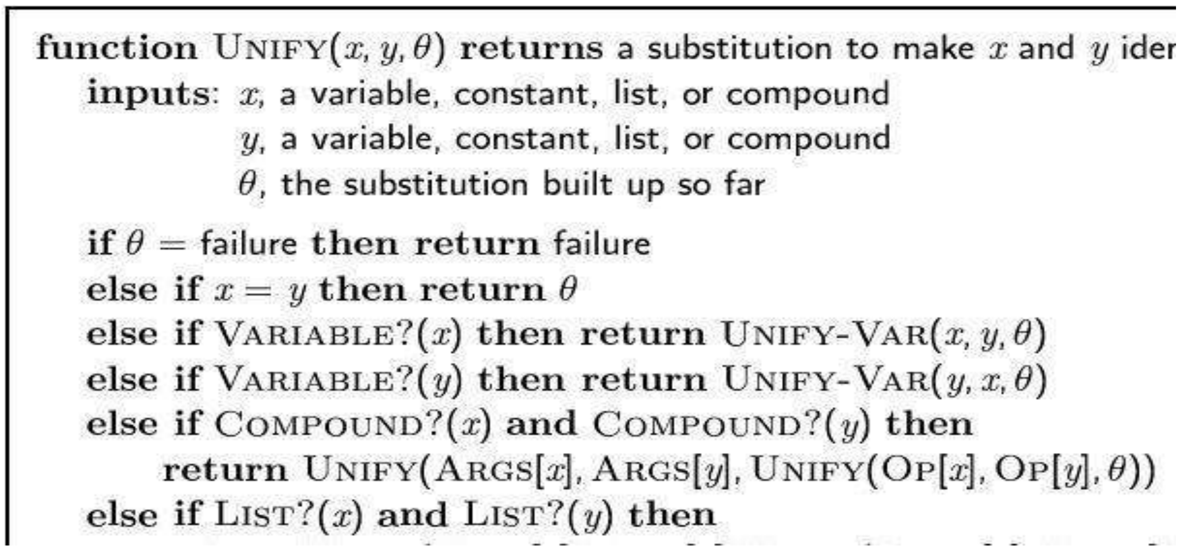
– Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ) To unify *Knows(John,x)* and *Knows(y,z)*, θ = {y/John, x/z } or θ = {y/John, x/John, z/John}

– The first unifier is more general than the second.

– There is a single most general unifier (MGU) that is unique up to renaming of variables.

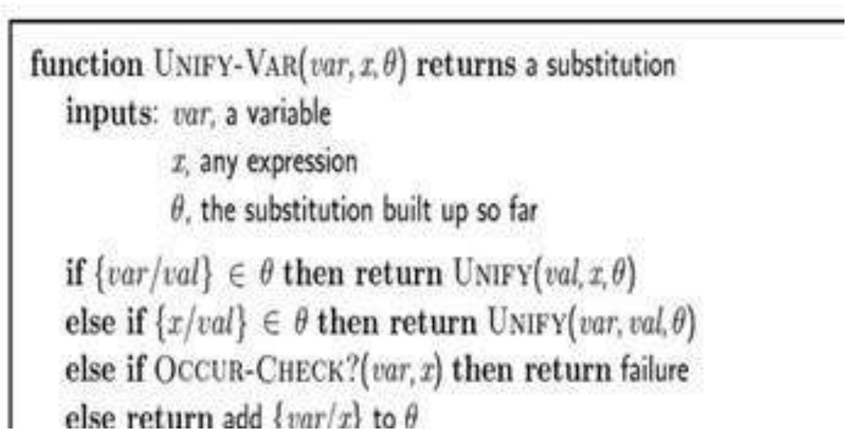
MGU = { y/John, x/z }

The unification algorithm



```
function UNIFY($x, y, \theta$) returns a substitution to make $x$ and $y$ iden
    inputs: $x$, a variable, constant, list, or compound
            $y$, a variable, constant, list, or compound
            $\theta$, the substitution built up so far

    if $\theta$ = failure then return failure
    else if $x = y$ then return $\theta$
    else if VARIABLE?($x$) then return UNIFY-VAR($x, y, \theta$)
    else if VARIABLE?($y$) then return UNIFY-VAR($y, x, \theta$)
    else if COMPOUND?($x$) and COMPOUND?($y$) then
        return UNIFY(ARGS[$x$], ARGS[$y$], UNIFY(OP[$x$], OP[$y$], $\theta$))
    else if LIST?($x$) and LIST?($y$) then
```



```
function UNIFY-VAR($var, x, \theta$) returns a substitution
    inputs: $var$, a variable
            $x$, any expression
            $\theta$, the substitution built up so far

    if $\{var/val\} \in \theta$ then return UNIFY($val, x, \theta$)
    else if $\{x/val\} \in \theta$ then return UNIFY($var, val, \theta$)
    else if OCCUR-CHECK?($var, x$) then return failure
    else return add $\{var/x\}$ to $\theta$
```

Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}{q\theta}$$

p_1' is *King*(*John*) p_1 is *King*(x)

p_2' is *Greedy*(y) p_2 is *Greedy*(x)

θ is {x/John,y/John} q is *Evil*(x) ``
q θ is *Evil*(*John*)

- GMP used with KB of definite clauses (exactly one positive literal)

- All variables assumed universally quantified

Soundness of GMP

- Need to show that

$p_1', \ldots, p_n', (p_1 \land \ldots \land p_n \Rightarrow q)$
$\models q\theta$ provided that $p_i'\theta$
$= p_i\theta$ for all *I*

- Lemma: For any sentence *p*, we have *p* $\models p\theta$ by UI

- $(p_1 \land \ldots \land p_n \Rightarrow q) \models (p_1 \land \ldots \land p_n \Rightarrow q)\theta = (p_1\theta \land \ldots \land p_n\theta \Rightarrow q\theta)$

- p_1', \; ..., \;$p_n' \models p_1' \land \ldots \land p_n' \models p_1'\theta \land \ldots \land p_n'\theta$

- From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

6. What is Resolution? Explain natural deduction(Nov'13)

Rules used in the resolution procedure are:

Simple resolution inference rule: premises have exactly two

disjuncts. (A V B, 7B V C) / (A V C)

(7A ➔ B, B ➔ C) / (7A ➔ C)

Resolution inference rule: two disjunctions of any length, if one of the disjunct in the clauses (Pj) unifies with the negation of the disjunct in the other class, then the disjunction of all the disjuncts except for those two.

Using disjunctions: for literals Pi and Qi, where

UNIFY (Pj , 7Qk) = θ;

P1 V……..Pj……..VPm, Q1V……..Qk VQn

SUBSET (0, (P1V………..Pj-1 V Pj+1……… Pm V Q1…………Qk-1 V Qk+1…………VQn))

using implications: for atoms Pi, Qi, Ri, Si where

UNIFY (pj , Qk) = θ

P1 ∧ ………..Pj………Pn1 ➔ r1V………rn2

S1∧ …………Sn3 ➔ Q1V………Qk…………Qn4

CANONICAL FORM (OR) NORMAL FORM FOR RESOLUTIONS:

The canonical form representation of sentences for resolution procedure is done in two ways. They are conjunction normal form (CNF): all the disjunctions are joined as one big sentence.

Implicative normal form (INF): all the conjunction of atoms on the left and a disjunction of atoms on the right

RESOLUTION TECHNIQUE (FROM FOL REPRESENTATION):

- The given KB description is converted into FOL sentences.
- FOL sentences are converted to INF (or) CNF representation without changing the meaning of it.
- Apply one of the resolution techniques to resolve the conflict.
- Derive the fact to be proved or declared it as an incomplete solution.

CONVERSION OF NORMAL FORM OR CLAUSE FORM:

- Elimination implications
- Move 7 inwards
- Move quantifiers left
- Skimolize
- Distribute ∧ over V
- Flatten nested conjunctions and disjunctions
- Convert disjunctions to implications.

7. Explain Weak slot - filler structure (Apr-May'14)(Apr-May'17) (APR-MAY'16)

We have seen using rules for knowledge representation. While rules have been widely used, there are several other important approaches to knowledge representation.

Three of these are

– *Structured objects*

– *Semantic nets*.

– *frames*;

– *logic*;

Structured Objects

Structured objects are: Knowledge representation formalisms whose components are essentially similar to the nodes and arcs found in graphs. In contrast to production rules and formal logic. An attempt to incorporate certain desirable features of human memory organization into knowledge representations.

• History of semantics networks (Quillian, 1968)

– To represent semantics of natural language words by dictionary-like definitions in a graphic form

– Defining the meaning of a word in terms of its relations with other words

– Semantic networks were very popular in the 60's and 70's and enjoy a much more limited use today.

– The graphical depiction associated with a semantic network is a big reason for their popularity.

A semantic (or associative) network is a simple representation scheme which uses a graph of labeled nodes and labeled, directed arcs to encode knowledge.

– Labeled nodes: objects/classes/concepts.

– Labeled links: relations/associations between nodes

– Labels define the semantics of nodes and links

– Large # of node labels (there are many distinct objects/classes)

Small # of link labels (types of associations can be merged into a few)

Buy, sale, give, steal, confiscation, etc., can all be represented as a single relation of "transfer ownership" between recipient and donor

– Usually used to represent static, taxonomic, concept dictionaries

– Semantic networks are typically used with a special set of accessing procedures which perform "reasoning"

– e.g., inheritance of values and relationships

– Often much less expressive than other KR formalisms

Nodes and Arcs

– Nodes denote objects/classes

– Arcs define binary relationships between objects.

Representation in a Semantic Net

The physical attributes of a person can be represented as in following figure:

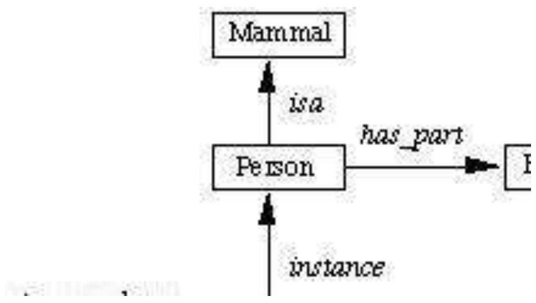


Fig. A Semantic Network

These values can also be represented in logic as: *is a(person, mammal), instance(Mike-Hall, person) team(Mike-Hall, Cardiff)*

We have already seen how conventional predicates such as *lecturer (dave)* can be written as *instance (dave, lecturer)* Recall that *is a* and *instance* represent inheritance and are popular in many knowledge representation schemes. But we have a problem:

How we can have more than 2 place predicates in semantic nets? E.g. score (Cardiff, Llanelli, 23-6)

Solution:

- Create new nodes to represent new objects either contained or alluded to in the knowledge, *game* and *fixture* in the current example.
- Relate information to nodes and fill up slots.

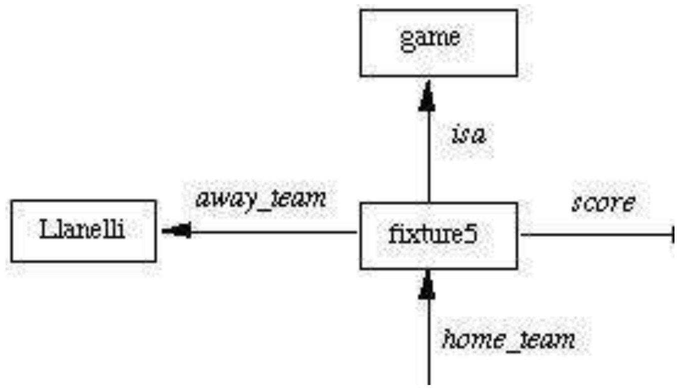


Fig. A Semantic Network for *n*-Place Predicate

As a more complex example consider the sentence: *John gave Mary the book*. Here we have several aspects of an event.

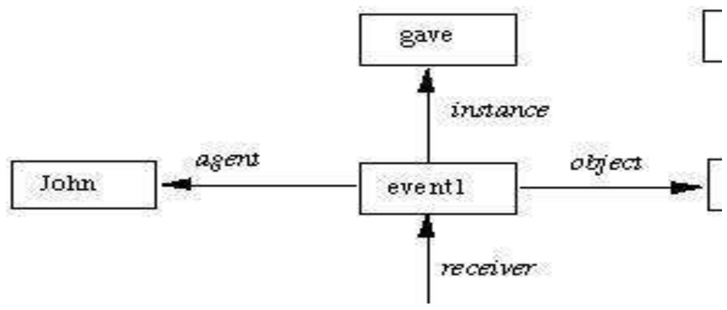


Fig. A Semantic Network for a Sentence

Inference in a Semantic Net

Basic inference mechanism: *follow links between nodes*.

Two methods to do this:

1. Intersection search

-- The notion that *spreading activation* out of two nodes and finding their intersection finds relationships among objects. This is achieved by assigning a special tag to each visited node.

Many advantages including entity-based organization and fast parallel implementation. However very structured questions need highly structured networks.

2. Inheritance

-- The *is a* and *instance* representation provide a mechanism to implement this.

Inheritance also provides a means of dealing with *default reasoning*. *E.g.* we could represent:

- Emus are birds.
- Typically birds fly and have wings.
- Emus run.

in the following Semantic net:

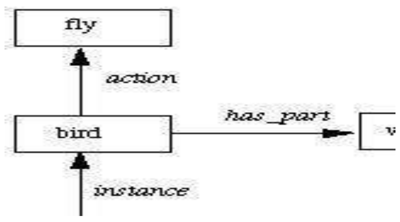


Fig. A Semantic Network for a Default Reasoning

In making certain inferences we will also need to *distinguish between the link that defines a new entity and holds its value and the other kind of link that relates two existing entities*. Consider the example shown where the height of two people is depicted and we also wish to compare them.

We need extra nodes for the concept as well as its value.

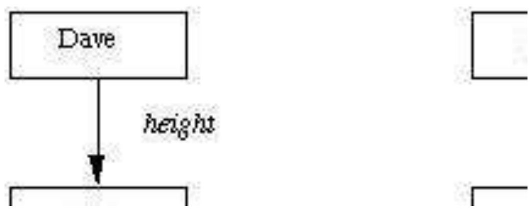


Fig. Two heights

Special procedures are needed to process these nodes, but without this distinction the analysis would be very limited.

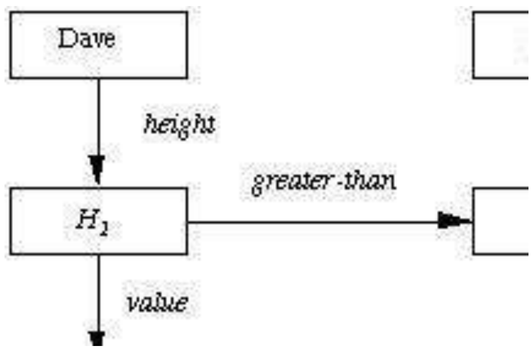


Fig. Comparison of two heights

E<u>xtending Semantic Nets</u>

Here we will consider some extensions to Semantic nets that overcome a few problems (see Exercises) or extend their expression of knowledge.

Partitioned Networks *Partitioned* Semantic Networks allow for:

– Propositions to be made without commitment to truth.

– Expressions to be quantified.

Basic idea: *Break network into spaces which consist of groups of nodes and arcs and regard each space as a node.* Nodes and arcs to link this *space* the rest of the network to represent Andrew's belief.

Consider the following: *Andrew believes that the earth is flat.* We can encode the proposition *the earth is flat* in a *space* and within it have nodes and arcs the represent the fact. We can have now consider the quantified expression: *Every parent loves their child* to represent this we:

- Create a *general statement*, GS, special class.
- Make node g an instance of GS.
- Every element will have at least 2 attributes:
    - a *form* that states which relation is being asserted.
    - one or more *for all* ($\forall$) or *exists* ($\exists$) connections -- these represent universally quantifiable variables in such statements *e.g. x*, *y* in

      $\forall x$:*parent(x)* $\rightarrow \exists y$: *child(y)* $\wedge$ *loves(x,y)*

Here we have to construct two *spaces one* for each *x*, *y*. NOTE: We can express ∃variables as *existentially qualified* variables and express the event of *love* having an agent *p* and receiver *b* for every parent *p* which could simplify the network

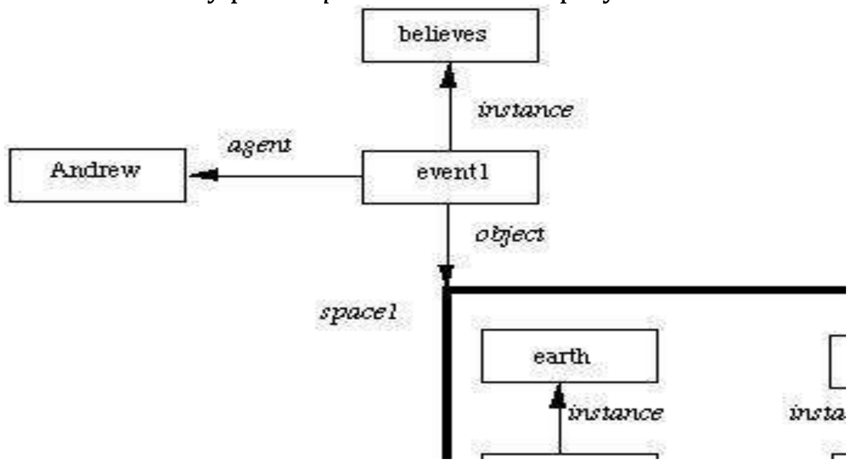
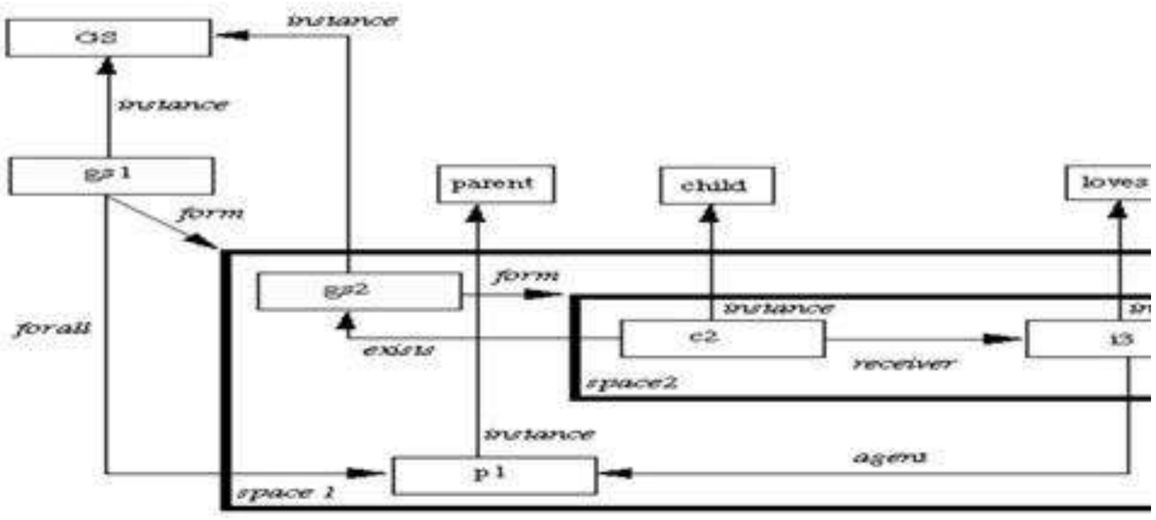


Fig. Partitioned network

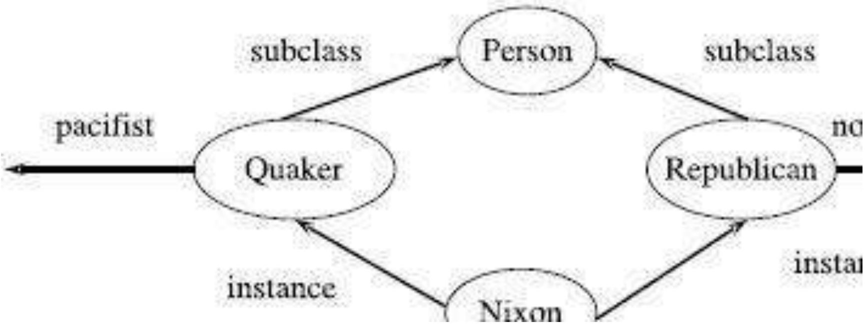
Also If we change the sentence to *Every parent loves child* then the node of the object being acted on (*the child*) lies outside the form of the general statement. Thus it is not viewed as an existentially qualified variable whose value may depend on the agent. (See Exercises and Rich and Knight Book for examples of this) So we could construct a partitioned network as in following figure:



Multiple Inheritance

A node can have any number of super classes that contain it, enabling a node to inherit properties from multiple parent nodes and their ancestors in the network. It can cause convicting inheritance.

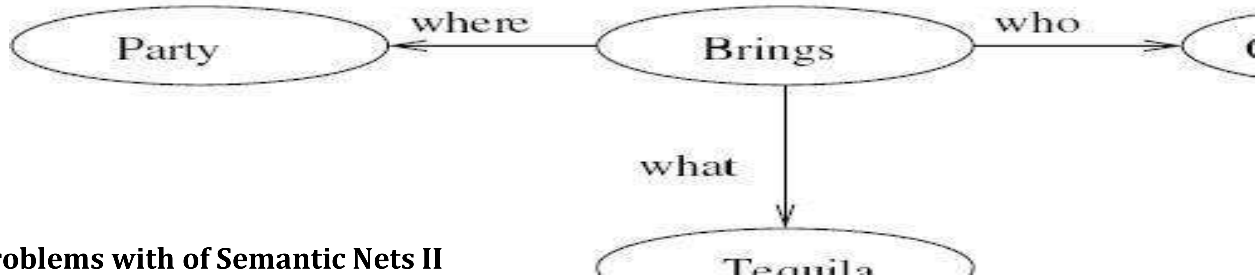
Nixon Diamond



Problems with Semantic Nets I

Binary relations are easy to represent. Others are harder.

Example: .Opus brings tequila to the party.



Problems with of Semantic Nets II

Other problematic statements. . .

– Negation .Opus does not ride a bicycle.;

– Disjunction .Opus eats oranges or apples.; *Quantized* statements are very hard for semantic nets.

Examples:

– Every dog has bitten a postman.

– Every dog has bitten every postman.

– *Partitioned* semantic nets can represent these.

Advantages of Semantic Nets

- o Easy to visualize
- o Relationships can be arbitrarily defined by the knowledge engineer
- o Formal definitions of semantic networks have been developed.
- o Related knowledge is easily clustered.
- o Efficient in space requirements

- Objects represented only once

Disadvantages of Semantic Nets

- Inheritance (particularly from multiple sources and when exceptions in inheritance are wanted) can cause problems.
- Facts placed inappropriately cause problems.
- No standards about node and arc values

Attempted Improvements

Building search heuristics into the network. More sophisticated logical structure, involving partitioning. These improvements meant that the formalism's -original simplicity was lost.

Frames

Frames. Semantic net with *properties* and *methods* Devised by Marvin Minsky, 1974. Incorporates certain valuable human thinking characteristics:

- Expectations, assumptions, stereotypes. Exceptions.
- Fuzzy boundaries between classes.
- The essence of this form of knowledge representation is *typicality*, with exceptions, rather than *definition*.

Hierarchical structure

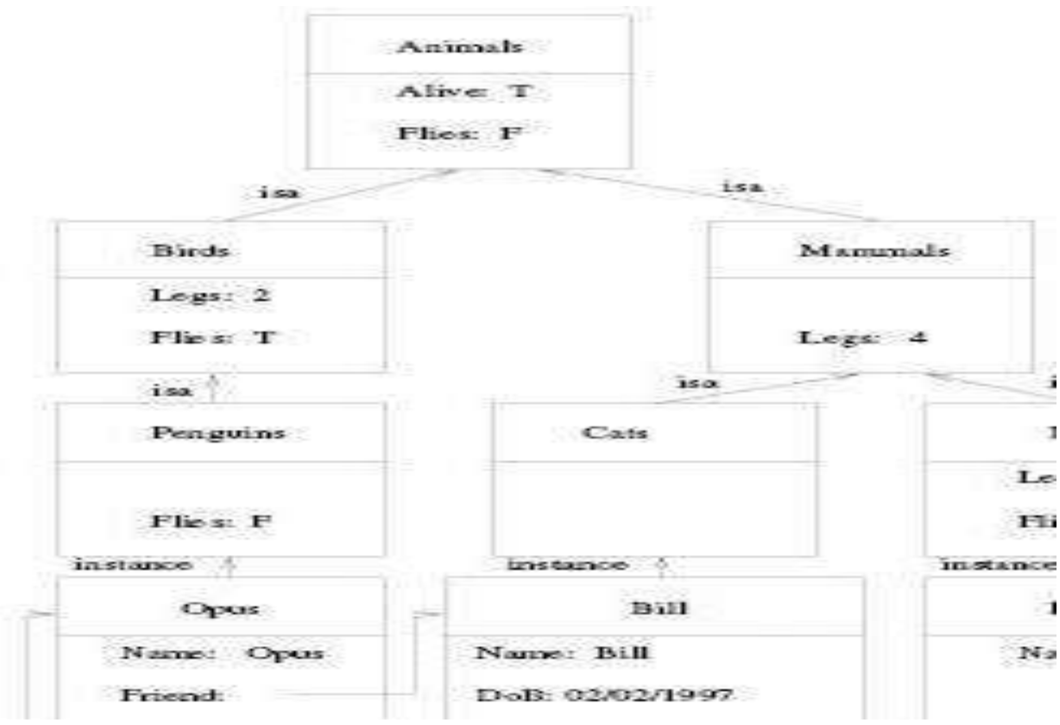
- Similar to class hierarchies

How Frames are Organized I

A frame can represent a specific entry, or a general concept each frame has

- A name
- Slots (attributes) which have *values* ○ a specific value
- a default value
- an inherited value
- a pointer to another frame
- a *procedure* that gives the value

Example frame system



Reasoning

- How to reason with frame systems?
- Easy to answer questions such as *is* x *a* y*?*
- Simply follow the instance and/or is a links.
- Example: Is Opus a bird?
- Also useful for *default* reasoning.
- Simply *inherit* all default values that are not explicitly provided.
- Example: How many legs does Opus have?

How Frames are Organized II

In the higher levels of the frame hierarchy, typical knowledge about the class is stored.
In the lower levels, the value in a slot may be a specific value, to overwrite the value which would otherwise be inherited from a higher frame. An instance of an object is joined to its class by an instance relationship. A class is joined to its superclass by is a relationship.

8. Explain Strong slot - filler structure.

Conceptual Dependency (CD)

Conceptual Dependency originally developed to represent knowledge acquired from natural language input.

The goals of this theory are:

- To help in the drawing of inference from sentences.

- To be independent of the words used in the original input.
- That is to say: *For any 2 (or more) sentences that are identical in meaning there should be only one representation of that meaning.*

It has been used by many programs that portend to understand English (*MARGIE, SAM, PAM*). CD developed by Schank *et al.* as were the previous examples.

CD provides:

- a structure into which nodes representing information can be placed
- a specific set of primitives
- at a given level of granularity.
- Sentences are represented as a series of diagrams depicting actions using both abstract and real physical situations.

- The agent and the objects are represented
- The actions are built up from a set of primitive acts which can be modified by tense.

Examples of Primitive Acts are:

ATRANS -- Transfer of an abstract relationship. *e.g. give*.

PTRANS -- Transfer of the physical location of an object. *e.g. go*.

PROPEL -- Application of a physical force to an object. *e.g. push*.

MTRANS-- Transfer of mental information. *e.g. tell*.

MBUILD -- Construct new information from old. *e.g. decide*.

SPEAK -- Utter a sound. *e.g. say*.

ATTEND-- Focus a sense on a stimulus. *e.g. listen, watch*.

MOVE -- Movement of a body part by owner. *e.g. punch, kick*.

GRASP-- Actor grasping an object. *e.g. clutch*.

INGEST-- Actor ingesting an object. *e.g. eat*.

EXPEL -- Actor getting rid of an object from body. *e.g. ????*.

Six primitive conceptual categories provide *building blocks* which are the set of allowable dependencies in the concepts in a sentence:

PP -- Real world objects.

ACT-- Real world actions.

PA -- Attributes of objects.

AA-- Attributes of actions.

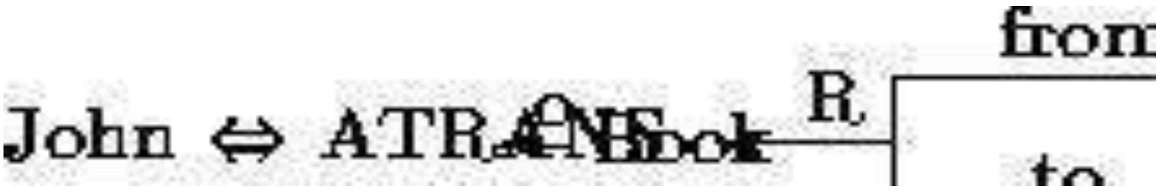
T -- Times.

LOC -- Locations.

How do we connect these things together?

Consider the example:

John gives Mary a book



Arrows indicate the direction of dependency. Letters above indicate certain

relationships: **o** -- object.

R -- recipient-donor.

I -- instrument *e.g. eat with a spoon*.

D -- destination *e.g. going home*.

- Double arrows (⇔) indicate *two-way* links between the actor (PP) and action (ACT).
- The actions are built from the set of primitive acts (see above).

- These can be modified by *tense etc.*

The use of tense and mood in describing events is extremely important and schank introduced the following modifiers:

p -- past

f – future

t -- transition

t_s -- start transition

t_f -- finished transition

k -- continuing

? -- interrogative

/ -- negative

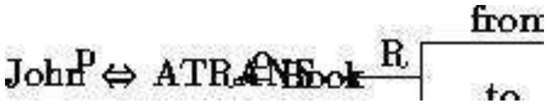
delta -- timeless

c -- conditional

the absence of any modifier implies the *present tense*.

So the *past tense* of the above example:

John gave Mary a book becomes:



The ⇔has an object (actor), PP and action, ACT. *I.e.* PP ⇔ACT. The triplearrow (⟺) is also a two link but between an object, PP, and its attribute, PA. *I.e.* PP ⟺PA.

It represents *isa* type dependencies. *E.g*

Dave ⟺lecturerDave is a lecturer.

Primitive states are used to describe many state descriptions such as height, health, mental state, physical state.

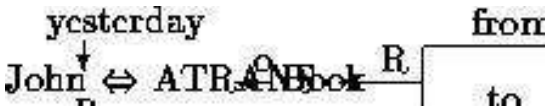
⟺

There are many more physical states than primitive actions. They use a numeric scale.

E.g. John ⟺height(+10) *John is the tallest* John height(< average) *John is short* Frank) Zappa ⟺health(-10) *Frank Zappa is dead* Dave ⟺ mental_state(-10) *Dave is sad* Vase ⟺physical_state(-10) *The vase is broken*

You can also specify things like the time of occurrence in the relationship.

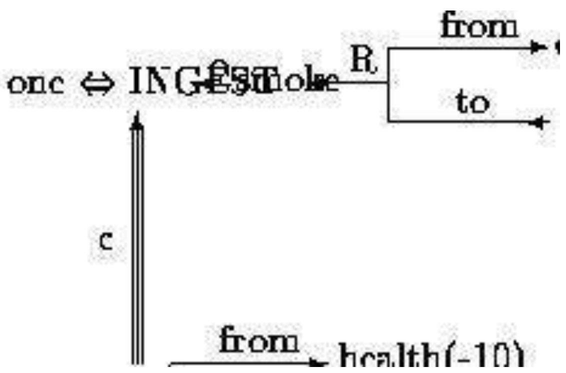
For Example: *John gave Mary the book yesterday*



Now let us consider a more complex sentence: *Since smoking can kill you, I stopped*

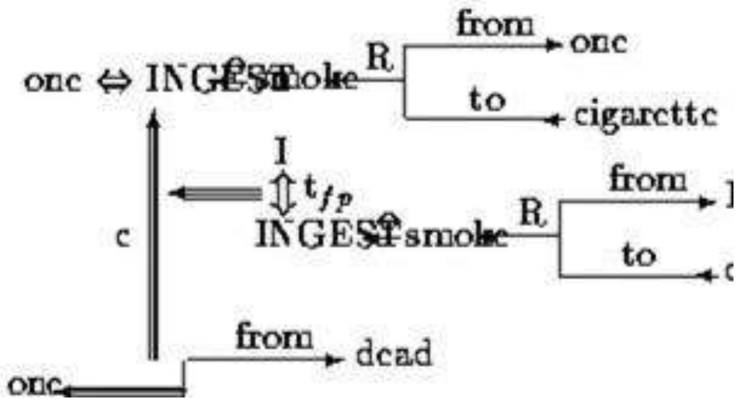
Lets look at how we represent the inference that *smoking can kill*:

– Use the notion of *one* to apply the knowledge to.
– Use the *primitive act* of INGESTing smoke from a cigarette to one.
– Killing is a transition from being alive to dead. We use *triple arrows* to indicate a transition from one state to another.
– Have a conditional, *c* causality link. The *triple arrow* indicates dependency of one concept on another.



To add the fact that *I stopped smoking*

- Use similar rules to imply that I smoke cigarettes.
- The qualification t_{fp} attached to this dependency indicates that the instance Ingesting smoke has stopped.



Advantages of CD:

- Using these primitives involves fewer inference rules.
- Many inference rules are already represented in CD structure.
- The holes in the initial structure help to focus on the points still to be established.

Disadvantages of CD:

- Knowledge must be decomposed into fairly low level primitives.
- Impossible or difficult to find correct set of primitives.

– A lot of inference may still be required.

– Representations can be complex even for relatively simple actions. Consider:

Dave bet Frank five pounds that Wales would win the Rugby World Cup.

Complex representations require a lot of storage

Applications of CD:

MARGIE (*Meaning Analysis, Response Generation and Inference on English*) -- model natural language understanding.

SAM (*Script Applier Mechanism*) -- Scripts to understand stories. See next section.

PAM (*Plan Applier Mechanism*) -- Scripts to understand stories.

QUESTION BANK & UNIVERSITY QUESTIONS

1. Explain the reasoning procedure using Fuzzy Logic.
2. Write about
    a. First Order Logic
    b. Rule Chaining
3. Explain semantic nets and positioned semantic nodes.
4. Write short notes on Prepositional Logic and First Order Logic (Nov-Dec'14)(DEC'17)(Pg. No 17, Q. No. 4)
5. Explain backtracking algorithm with example Represent the following sentences in first order logic using a consistent vocabulary. (Which you must define).
    a. Some students took French in spring 2001.
    b. Every student who takes French passes it.
    c. Only one student took Greek in spring 2001.
    d. The best score in Greek is always higher than the best score in French.
    e. Every person who buys a policy is smart.

f. No person buys an expensive policy.
g. There is an agent who sells policies only to people who are not insured.

7. (a) Explain in detail about Frames with an example.

   (b) Discuss forward chaining in detail.

8. (a) Describe Inference in First Order Logic in detail.
   (b) Explain backward chaining algorithm with suitable example.
9. What are the issues in knowledge representation? Explain.(Nov'13)(Apr-May'14)(Nov-Dec'15) (April 18) (Pg. No 10, Q. No. 1)

10. Explain the weak slot and filler structure.(Apr-May'14)(Pg. No 27, Q. No 7)

11. Explain in detail about Resolution and natural deduction(Nov'13)Nov – Dec'14)(Pg. No 25, Q. No. 6)

12. Explain the resolution in Propositional Logic with an example (Apr-May'15) (April 18) (Pg. No.15, Q. No. 3)

13. State the unification algorithm and describe in detail (Apr-May'15)(Pg. No.22, Q. No. 5)

14. Explain First Order Logic in Detail. (Nov'15)(Dec '17) (Pg. No 18, Q. No. 4)

15. Give resolution proof for example problem statement: (Nov '18)

    (a) "West is a criminal"

    (b) Curiosity killed the cat.

16. Explain forward chaining and backward chaining with an example. (Nov '18)